IFW AF 2124

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application: Krishna Sangavarapu

Serial No.: 09/732,250

Filed: December 7, 2000

Art Unit: 2124

Examiner: Insun Kang

For: A Method of Detecting Zombie Breakpoints

Attorney Docket: JP20000280US1

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Dear Sir:

REAL PARTY IN INTEREST

The assignee, International Business Machines Corporation, is the real party in interest.

RELATED APPEALS AND INTERFERENCES

This is the first appeal in the present patent application. There are no other appeals or interferences known to the appellant or its legal representative. International Business Machines Corporation is the sole assignee of the patent application.

08/18/2004 KBETEMA1 00000043 090457 09732250
01 FC:1401 330.00 DA

STATUS OF CLAIMS

Claims 1 through 18 stand finally rejected. Office action, March 10, 2004 (the "Final Office Action"). Appellant has appealed from the final rejection of claims 1 through 18. Notice of Appeal, received by USPTO on June 14, 2004.[1]

Claims 1 through 18 were originally presented in the application. Independent claims 1, 7 and 13 were amended in Reply A, filed January 26, 2004, in response to a first Office action, dated October 27, 2003, (the "First Office Action") so that the claims clearly point out patentable distinctions with regard to the art cited in the First Office action. No claims have been canceled, and no claims have been allowed.

STATUS OF AMENDMENTS

No amendments were filed subsequent to the Final Office Action. The claims set out herein in Appendix "AA" reflect the amendments as entered responsive to Appellant's reply to the First Office Action.

SUMMARY OF INVENTION

The present invention is claimed in the form of a method, an apparatus, and a computer program product in claims 1, 7 and 13, respectively, and concerns breakpoint handling in a multithreading processing environment. See present application, page 2, lines 11-23.

It is conventional that when a process encounters a breakpoint a debugger temporarily removes the breakpoint. An instruction for which the breakpoint was originally substituted is then executed, and then the breakpoint is immediately replaced so that if the instruction is again encountered the breakpoint will fire. The present invention deals with a problem that arises when two threads encounter the same breakpoint and the debugger temporarily removes the breakpoint for one of the threads while processing of the breakpoint for the second thread is still pending. The problem is that when the processing of the breakpoint occurs for the second thread the breakpoint may be absent because it has been temporarily removed for the

[1] Appellant notes that the USPTO Patent Application Information Retrieval data does not reflect the receipt by the PTO of Appellant's Notice of Appeal. Consequently, a copy is herein provided in Appendix "BB" of the Notice and an acknowledged postcard as evidence that the Notice of Appeal was timely filed.

first thread. The absent breakpoint is referred to in the present application as a "zombie" breakpoint. The above described problem will be referred to herein as the "zombie breakpoint problem."

According to the present invention, a breakpoint is *temporarily removed* from a line of code upon encountering the breakpoint so that the instruction for which the breakpoint was substituted can be replaced and executed. Present application, page 6, line 10 through page 7, line 1. Consequently, claim 1 states that a breakpoint data structure is checked to determine if the data structure has an entry for a breakpoint known to a debugging process for a certain address where a breakpoint fired. (Herein claim 1 is discussed. However, it should be understood independent claims 7 and 13 have similar language, each according to the form of the invention they claim.) Then, the claim goes on, there is a step of verifying if a breakpoint condition continues to exist at the address where the breakpoint fired if no entry is found by the step of checking the data structure for the known breakpoint. Present application page 5, lines 24 through 30.

Thus, in the first step the checking looks in one place, the data structure, and in the second step the verifying looks somewhere else, which could be the actual memory address for the breakpoint or possibly in a breakpoint register. See present application, page 5, line 34 - page 6, line 5. Claim 1 then states that "if said breakpoint does not exist," i.e., because it has been temporarily removed, the breakpoint is identified as a zombie breakpoint. This is contrasted to the conventional result, according to which it is incorrectly concluded, due to the absence of the removed breakpoint, that the exception was not caused by a breakpoint. Present application, page 5, lines 18 through 20.

In addition to the above discussed independent claims 1, 7 and 13, the present application has a number of dependent claims, 2-6, 8-12 and 14-18. Since the broadest, independent claims are patentably distinct, the dependent claims are likewise patentably distinct merely based on their dependence upon the respective independent claims. Moreover, the dependent claims 2-6, 8-12 and 14-18 are all the more patentably distinct due to their respective additional limitations.

ISSUES

Are claims 1 through 18 unpatentable under 35 U.S.C. 102(e) as being anticipated by U.S. Patent 6,484,818 ("Alverson")?

GROUPING OF CLAIMS

Solely for the purpose of this appeal, claims 1-18 stand or fall together.

ARGUMENT

The Final Office Action contends claims 1 through 18 are unpatentable under 35 U.S.C. 102(e) as being anticipated by Alverson. Appellant respectfully disagrees. All the words of a claim must be considered in a rejection pursuant to 35 U.S.C. 102. MPEP 2131 (citing *Verdegaal Bros. v. Union Oil Co., of California*, 814 F.2d 628, 631 ("A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference.")). Alverson does not teach or even suggest all the elements set forth in the claims of the present application.

Alverson does not teach the same method and structure for solving the zombie breakpoint problem. Alverson, like the present invention, concerns a debugger for controlling execution of target code for which there are multiple thread processes. Alverson, col. 9, lines 15 through 17. Alverson recognizes and deals with the same zombie breakpoint problem as the present invention. Alverson, col. 13, lines 3 through 10. However, Alverson handles this problem in a different manner, and thus does not teach or suggest the present claimed invention.

Appellant has considered the teaching of Alverson referred to by the Office action, and offers the following by way of summary in order to place the cited teaching into context. Alverson advocates and teaches about a debugger that has a root nub and individual nubs for the respective target thread process. Alverson, col. 9, lines 5 through 6. The debugger nubs obtain state information about their respective target thread processes. Alverson, col. 9, lines 28 through 30. When an exception occurs, a general trap handler determines the type of execution causing the trap so that the general trap handler can invoke the right exception trap handler. Alverson, col. 10, lines 37 through 39. If the trap is caused by a breakpoint then a

breakpoint exception handler can be invoked. The breakpoint exception handler can interact with the debugger nub thread for the target thread process that gave rise to firing of the breakpoint, in order to allow debugger control. Alverson, col. 10, lines 44 through 54. The debugger nub can get information about the target thread from a save area. Id.

Alverson indicates that it is known to use a domain signal to manipulate all the threads in a system. Alverson, col. 11, lines 31 through 33. Alverson teaches that the debugger nubs may need to ignore the domain signal. Alverson, col. 11, lines 33 through 35. Also, a target thread process may need to temporarily ignore the domain signal while its nub accesses the target's data structure. Alverson, col. 11, lines 37 through 41.

More to the point of the present invention, Alverson offers a solution to the zombie breakpoint problem, as follows. When a breakpoint is inserted for an instruction, the debugger generates an out of line instruction emulation group so that the instruction for which the breakpoint is substituted can be executed in another area of memory, that is, "out of line. " Alverson, col. 13, lines 11 through 17; see also FIG's 4A and 4B. Instead of temporarily replacing the instruction back in line after encountering the breakpoint, as is conventionally done, the breakpoint handler transfers execution to the instruction emulation group. Alverson, col. 13, lines 43 through 48. After executing the instruction in the instruction emulation group, that is, after executing the instruction for which the breakpoint was substituted, the target thread resumes execution with the next instruction in the original, in-line code. Alverson, col. 14, lines 14 through 17. Thus, by generating the out of line instruction emulation group and executing the substituted instruction in the instruction emulation group instead of temporarily replacing the instruction back in-line, "the processing of the breakpoint has been performed *without removing the BREAK instruction from the target code instructions*. Thus if another target thread had executed the same instructions while the breakpoint for the first target thread was being processed, the second target thread will also encounter the breakpoint instead of inadvertently missing a temporarily absent BREAK instruction." Alverson, col. 14, lines 31 through 38 (emphasis added). This teaching by Alverson is directly contrary to that of the present invention and does not teach or even suggest what is claimed in the present application.

Applicant's remarks in the Reply to first Office action focused primarily on Alverson's teachings about what Alverson calls "out-of-line" processing of breakpoints by instruction emulation, because it appeared to Appellant that this is the teaching propounded by Alversan as new. The Final Office Action, however, relies on Alverson, col. 17, lines 20-40, for teaching about in-line processing of breakpoints. As discussed with the Examiner in an Appellant initiated interview after final rejection, this passage of Alverson, and particularly Alverson col. 17, lines 24-30, teaches that if the out-of-line instruction emulation techniques taught by the Alverson invention cannot be practiced because some permission does not exist to permit emulation of an instruction, then breakpoints must be handled in-line, i.e., without the use of out-of-line instructions and emulation. The choices Alverson teaches for in-line handling of breakpoints are either a) threads have to be halted or b) it must just be accepted that some thread might miss a breakpoint. These choices are nothing more than references to the zombie breakpoint problem addressed by the present invention, and neither choice offers a solution to the problem that anticipates the present invention.

Alverson offers no solution at all in connection with choice b). Missing a breakpoint clearly does not anticipate the present claimed invention, because the claims in the present case state that if a breakpoint does not exist, i.e., a breakpoint has been temporarily removed so that the instruction replaced by the breakpoint can be executed, then the breakpoint is identified. See, e.g., present application, claim 1 (last step). Since the breakpoint is identified it is decidedly *not* missed, as in Alverson's choice b).

With regard to choice a), halting threads, Alverson does not actually teach what this means. The mere statement that threads may be halted in connection with breakpoints clearly does not anticipate the present claimed invention. Even if Alverson's alluding to the halting of threads is analyzed in search for an *implied suggestion* that *might* relate to the present invention, such implication is different than what is taught and claimed in the present application.

The zombie breakpoint problem is fundamentally a multithreading problem. See present application, page 2, lines 11-23. Alverson is merely suggesting to halt all other threads whenever one of the threads encounters a breakpoint until such time as the breakpoint for the first thread has been handled and put back into the instruction stream. In this manner, a second

thread *cannot* encounter a breakpoint concurrently with processing of the breakpoint by a debugger for the first thread. Consequently, the zombie breakpoint problem never arises.

This implied suggestion of Alverson is contrary to the teaching in the present application. The claims and the specification in the present case make it clear that the present invention is not about merely resorting to single thread processing. All other threads are *not* halted when one thread encounters a breakpoint. That is, the specification states that threads A and B hit a breakpoint at nearly the same time and a debugger sequentially processes the threads. Present application page 5, lines 6-14. If the debugger temporarily removes the breakpoint for one of the threads while processing of the breakpoint for the second thread is still pending, then when the debugger processes the second thread the breakpoint may be absent, i.e., the breakpoint may be "missed," because it has been temporarily removed for the first thread. Id. Thus according to the teaching of the present invention, the problem of a zombie breakpoint is handled in the *multithreading* environment. Present application page 2, lines 12-14. The claims of the present application are directed to how this is handled.

Claim 1, for example, states that first a breakpoint data structure is checked to determine if the data structure has an entry for a breakpoint known to a debugging process for a certain address where a breakpoint fired. Then, the claim goes on, if no entry is found by the step of checking the data structure for the known breakpoint, the method verifies whether a breakpoint condition continues to exist at the address where the breakpoint fired. If the breakpoint condition does not still exist, i.e., the breakpoint condition is missing, then the method identifies the breakpoint as a zombie breakpoint. The breakpoint condition would not be missing, as indicated in claim 1 of the present application, if other threads were halted upon any one of the threads encountering a breakpoint.

From the above it should be appreciated that Alverson does not anticipate the claimed invention. Neither of the choices offered by Alverson for in-line handling of breakpoints teach that an absent breakpoint is identified as a zombie breakpoint by checking a breakpoint data structure to determine if the data structure has an entry for a breakpoint known to a debugging process for a certain address where a breakpoint fired, and then, if no entry is found by the checking, verifying whether a breakpoint condition continues to exist at the address where the breakpoint fired, as claimed in the present application.

REQUEST FOR ACTION

Based on the above arguments, Appellant requests that claims 1 through 18 of present application be allowed and the application promptly passed to issuance.

Respectfully submitted,

By Anthony VS England

Anthony V.S. England
Registration No. 35,129
Attorney of Record for
IBM Corporation
1717 West Sixth Street, Suite 230
Austin, Texas 78703
Telephone: 512-477-7165
a@aengland.com

Attachment: Appendix "AA" Claims
Appendix "BB" Notice of Appeal and acknowledged postcard

What is claimed is:

1. (previously presented) A method of detecting one or more zombie global breakpoints for debugging computer software, said method including the steps of:

checking a breakpoint data structure to determine if the data structure has an entry for a breakpoint known to a debugging process for a certain address where a breakpoint fired;

if no entry is found by the checking of the data structure for the entry for the known breakpoint, verifying if a breakpoint condition continues to exist at the address where the breakpoint fired; and

if said breakpoint condition does not exist, identifying said breakpoint as a zombie breakpoint.

2. (original) The method according to claim 1, wherein said verifying step includes the step of checking that a special breakpoint instruction exists at said address, being the exception location.

3. (original) The method according to claim 1, wherein said verifying step includes the step of checking that an illegal breakpoint instruction exists at said address, being the exception location.

4. (original) The method according to claim 1, wherein said verifying step includes the step of checking that said address, being the exception location, is present in a special debug register.

5. (original) The method according to claim 1, wherein physical settings for causing a breakpoint exception at a particular location are detectable from a breakpoint handler.

6. (original) The method according to claim 5, wherein breakpoint removal logic is provided that lifts a physical breakpoint instruction from a breakpoint location before removing a breakpoint entry from said breakpoint data structure of said debugging process.

7. (previously presented) A computer-implemented apparatus for detecting one or more zombie global breakpoints for debugging computer software, said apparatus including:

a central processing unit for executing computer software;

memory for storing at least a portion of said computer software;

means for checking a breakpoint data structure to determine if the data structure has an entry for a breakpoint known to a debugging process for a certain address where a breakpoint fired;

means for, if no entry is found by the checking of the data structure for the entry for the known breakpoint, verifying if a breakpoint condition continues to exist at the address where the breakpoint fired; and

means for, if said breakpoint condition does not exist, identifying said breakpoint as a zombie breakpoint.

8. (original) The apparatus according to claim 7, wherein said verifying means includes means for checking that a special breakpoint instruction exists at said address, being the exception location.

9. (original) The apparatus according to claim 7, wherein said verifying means includes means for checking that an illegal breakpoint instruction exists at said address, being the exception location.

10. (original) The apparatus according to claim 7, wherein said verifying means includes means for checking that said address, being the exception location, is present in a special debug register.

11. (original) The apparatus according to claim 7, wherein physical settings for causing a breakpoint exception at a particular location are detectable from a breakpoint handler.

12. (original) The apparatus according to claim 11, wherein breakpoint removal logic is provided that lifts a physical breakpoint instruction from a breakpoint location before removing a breakpoint entry from said breakpoint data structure of said debugging process.

13. (previously presented) A computer program product having a computer readable medium having a computer program recorded therein for detecting one or more zombie global breakpoints for debugging computer software, said computer program product including:

computer program code means for checking a breakpoint data structure to determine if the data structure has an entry for a breakpoint known to a debugging process for a certain address where a breakpoint fired;

computer program code means for, if no entry is found by the checking of the data structure for the entry for the known breakpoint, verifying if a breakpoint condition continues to exist at the address where the breakpoint fired; and

computer program code means for, if said breakpoint condition does not exist, identifying said breakpoint as a zombie breakpoint.

14. (original) The computer program product according to claim 13, wherein said computer program code means for verifying includes computer program code means for checking that a special breakpoint instruction exists at said address, being the exception location.

15. (original) The computer program product according to claim 13, wherein said computer program code means for verifying includes computer program code means for checking that an illegal breakpoint instruction exists at said address, being the exception location.

16. (original) The computer program product according to claim 13, wherein said computer program code means for verifying includes computer program code means for checking that said address, being the exception location, is present in a special debug register.

17. (original) The computer program product according to claim 13, wherein physical settings for causing a breakpoint exception at a particular location are detectable from a breakpoint handler.

18. (original) The computer program product according to claim 17, wherein breakpoint removal logic is provided that lifts a physical breakpoint instruction from a breakpoint location before removing a breakpoint entry from said breakpoint data structure of said debugging process.

PTO/SB/31 (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

NOTICE OF APPEAL FROM THE EXAMINER TO THE BOARD OF PATENT APPEALS AND INTEFERENCES

Docket Number (Optional): JP92000280US1

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to "Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450" on 6/9/2004

Signature: Anthony V.S. England

Typed or printed name: Anthony V.S. England

In re Application of: Vamsi Krishna Sangavarapu

Application Num: 09/732,250 | Filed: 12/07/2000

For: A METHOD OF DETECTING ZOMBIE BREAKPOINTS

Art Unit: 2124 | Examiner: Insun Kang

Applicant hereby **appeals** to the Board of Patent Appeals and Interferences from the last decision of the examiner.

The fee for this Notice of Appeal is (37 CFR 1.17(b)) $ 330.00

☐ Applicant claims small entity status. See 37 CFR 1.27. Therefore, the fee shown above is reduced by half, and the resulting fee is: $ _____

☐ A check in the amount of the fee is enclosed.

☐ Payment by credit card. Form PTO-2038 is attached.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account. I have enclosed a duplicate copy of this sheet.

☒ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. 09-0457 . I have enclosed a duplicate copy of this sheet.

☐ A petition for an extension of time under 37 CFR 1.136(a) (PTO/SB/22) is enclosed.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

I am the

☐ applicant/inventor.

☐ assignee of record of the entire interest. See 37 CFR 3.71. Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96)

☒ attorney or agent of record. Registration number 35,129.

☐ attorney or agent acting under 37 CFR 1.34(a). Registration number if acting under 37 CFR 1.34(a). _____

Anthony V.S. England
Signature

Anthony V.S. England
Typed or printed name

512-477-7165
Telephone number

6-9-2004
Date

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below*.

☒ *Total of 1 forms are submitted.

Docket No.: JP920000280US1 Date Mailed: 6/9/04
PLEASE ACKNOWLEDGE AND RETURN

___ Certificate of Express Mail No.:
___ Transmittal Letter
___ Specification (___ pages)
___ Declaration and Power of Attorney
___ Drawings
___ Recordation Cover Sheet and Assignment
___ IDS and copies of non US patent references
X Other Notice of Appeal

Applicant: Sangavarapu
Title: A Method of Detecting Zombie Peak Points

PTO/SB/17 (10-03)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

FEE TRANSMITTAL for FY 2004

Effective 10/01/2003. Patent fees are subject to annual revision.

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) 330.00

Complete if Known	
Application Number	09/732,250
Filing Date	12/07/2000
First Named Inventor	Vamsi Krishna Sangavarapu
Examiner Name	Insun Kang
Art Unit	2124
Attorney Docket No.	JP920000280US1

METHOD OF PAYMENT *(check all that apply)*

☐ Check ☐ Credit card ☐ Money Order ☐ Other ☐ None

☑ Deposit Account:

Deposit Account Number: 09-0457

Deposit Account Name: International Business Ma

The Director is authorized to: *(check all that apply)*

☑ Charge fee(s) indicated below ☑ Credit any overpayments

☑ Charge any additional fee(s) or any underpayment of fee(s)

☐ Charge fee(s) indicated below, **except for the filing fee** to the above-identified deposit account.

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1001	770	2001	385	Utility filing fee	
1002	340	2002	170	Design filing fee	
1003	530	2003	265	Plant filing fee	
1004	770	2004	385	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	

SUBTOTAL (1) ($) 0

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

	Extra Claims		Fee from below	Fee Paid
Total Claims		-20** =		X =
Independent Claims		- 3** =		X =
Multiple Dependent				=

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
1202	18	2202	9	Claims in excess of 20
1201	86	2201	43	Independent claims in excess of 3
1203	290	2203	145	Multiple dependent claim, if not paid
1204	86	2204	43	** Reissue independent claims over original patent
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 0

***or number previously paid, if greater; For Reissues, see above*

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet	
1053	130	1053	130	Non-English specification	
1812	2,520	1812	2,520	For filing a request for *ex parte* reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action	
1251	110	2251	55	Extension for reply within first month	
1252	420	2252	210	Extension for reply within second month	
1253	950	2253	475	Extension for reply within third month	
1254	1,480	2254	740	Extension for reply within fourth month	
1255	2,010	2255	1,005	Extension for reply within fifth month	
1401	330	2401	165	Notice of Appeal	
1402	330	2402	165	Filing a brief in support of an appeal	$330.00
1403	290	2403	145	Request for oral hearing	
1451	1,510	1451	1,510	Petition to institute a public use proceeding	
1452	110	2452	55	Petition to revive - unavoidable	
1453	1,330	2453	665	Petition to revive - unintentional	
1501	1,330	2501	665	Utility issue fee (or reissue)	
1502	480	2502	240	Design issue fee	
1503	640	2503	320	Plant issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	
1809	770	2809	385	Filing a submission after final rejection (37 CFR 1.129(a))	
1810	770	2810	385	For each additional invention to be examined (37 CFR 1.129(b))	
1801	770	2801	385	Request for Continued Examination (RCE)	
1802	900	1802	900	Request for expedited examination of a design application	

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($) 330.00

SUBMITTED BY

(Complete *(if applicable)*)

Name *(Print/Type)*	Anthony V.S. England	Registration No. *(Attorney/Agent)*	35,129	Telephone	512-477-7165
Signature	Anthony V.S. England			Date	8/15/2004

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

PTO/SB/21 (08-03)
Approved for use through 08/30/2003. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/732,250
Filing Date	12/07/2000
First Named Inventor	Vamsi Krishna Sangavarapu
Art Unit	2124
Examiner Name	Insun Kang
Total Number of Pages in This Submission: 47	Attorney Docket Number: JP920000280US1

ENCLOSURES *(Check all that apply)*

- [x] Fee Transmittal Form
  - [x] Fee Attached
- [ ] Amendment/Reply
  - [ ] After Final
  - [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Response to Missing Parts/ Incomplete Application
  - [ ] Response to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition to Convert to a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s) _____
- [ ] After Allowance communication to Technology Center (TC)
- [ ] Appeal Communication to Board of Appeals and Interferences
- [x] Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)***
- [ ] Proprietary Information
- [ ] Status Letter
- [x] Other Enclosure(s) (please Identify below):

Acknowledgement postcard

Remarks

*in triplicate (Appeal Brief 8 pages, Appendix "AA" 4 pages, Appendix "BB" 3 pages)

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	Anthony V.S. England
Signature	Anthony V S England
Date	8/15/2004

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below.

Typed or printed name	Anthony V.S. England		
Signature	Anthony V S England	Date	8/15/2004